TAMPERE UNIVERSITY OF TECHNOLOGY

KARTHIKEYAN GANESAN

Physical activity and event detection with inertial sensors: Design and evaluation for low power microcontroller

Master of Science Thesis

Examiners: Prof.Irek Defee
Prof.Mikko Valkama
Examiners and Topic approved in the
Faculty Council Meeting on
11.01.2012

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
KARTHIKEYAN GANESAN : Physical activity and event detection with inertial sensors: Design and Evaluation for low power MCU
Master of Science Thesis, 42 pages, 1 Appendix pages
May 2012
Major: Communication Engineering
Examiner: Prof. Irek Defee, Prof. Mikko Valkama
Keywords: Sensor processor, Bluetooth, Physical activity context, Free-fall detection

The aim of this thesis is to design inertial sensor based activity recognition for a low power continuous sensing architecture for today's mobile technology using a dedicated low-power sensor processor. The system framework is designed to provide higher level access to sensors and run sensor based algorithms. The sensor processor can provide consistently low latencies for feedback and interaction and continuous low power sensing for context adaptive user interfaces. In the current state of the art of hand-helds, every sensor is interfaced to the host processor directly and all the sensory data preprocessing and event detection is performed at the expense of the host processor. A dedicated sensor processor with good system architecture would improve the power and processing efficiency.

The objective of the thesis is to interface all inertial sensors to a low-power discrete sensor hub where this low power discrete sensor hub implements most of the sensor pre-processing and algorithm. This sensor hub is directly interfaced via Bluetooth link to the host processor which is an Android mobile device. The sensor firmware architecture is event driven and reduces the power consumption without the need for polling the sensors. Host processor sends control messages to the sensor box and the device when replies back to the control messages also produces sensor data so most of the traffic is from the sensor hub to the host.

Activity context detects user actions such as stand-still, walking, running, cycling, vehicle and sports. The activity context module uses naive bayesian classifier algorithm to classify the accelerometer sensor data and this module is implemented in the sensor box which has an ARM based microcontroller. Sensor drivers and sensor algorithms are tested using a test framework implemented in the application program. Event detection module implements free-fall detection in the sensor processor with the help of accelerometer data which shows the capability of the system. Classification algorithm verified in simulation environment shows around 92 percent accuracy in detecting the physical activity of the user.

PREFACE

I would like to express my sincere thanks and appreciation to Prof. Irek Defee, for giving me the opportunity to work at the Department of Signal Processing and for his attention, supervision, support and comments during this research and the preparation of this thesis. I would also like to sincerely thanks Prof. Mikko Valkama for his comments during the preparation of this thesis. This thesis is done based on the advise, guidance, invaluable insights and knowledge shared by my advisor MSc. Rajasekaran Andiappan. I would like to gratefully thank him for sharing his knowledge with me. I also would like to thank Nokia Research Center for providing the necessary hardware for the work. I also wish to thank Mr. Peter Adamondy for his contribution to this project. My final words go to my parents. I want to thank my parents, whose love and guidance is with me in whatever I pursue.

Karthikeyan Ganesan

May 2012

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

OPERATORS, SYMBOLS AND NOTATION

$*$	Convolution
$\times$	Multiplication
$\sum_a^b$	Sum from a to b
F_{lim}	Nyquist frequency, which is equal to $F_S/2$
F_S	Sampling frequency
Hz	Hertz = number of cycles per second, unit of frequency
m	meters
ms	milliseconds
mA	milliampere

ABBREVIATIONS AND NOTATION

MCU	Microcontroller
CPU	Central Processing Unit
RAM	Random Access Memory
OS	Operating System
USB	Universal Serial Bus
I2C	Inter-Integrated Circuit
GPS	Global Positioning System
WLAN	Wireless Local Area Network
LAN	Local Area Network
BT	Bluetooth
SDK	Software Development Kit
NDK	Native Development Kit
RISC	Reduced Instruction Set Computing
ARM	Advanced RISC Machines
MARG	Magnetic, Angular Rate, and Gravity
IMU	Inertial Motion Unit
INS	Inertial Navigational System
FFT	Fast Fourier Transform
VAR	Variance
STD	Standard Deviation
ZCR	Zero Crossing Rate
DFT	Discrete Fourier Transform
PSD	Power Spectral Density

1. INTRODUCTION

During the last few years there is a shift of focus from laptops, notebooks to pocket sized hand-helds. Nowadays mobile devices are abundant and so cheap that people carry and use this device wherever they go. The impact and usefulness of these devices are not fully explored yet which creates scope for improvement. The aim of this thesis is to design a low power continuous sensing architecture for today's mobile technology using a dedicated low-power sensing processor for sampling and processing of sensor data. This processor is typically a microcontroller consisting of a CPU, RAM, and various peripherals such as direct memory access, analog to digital converter, serial communication buses like I2C etc. This dedicated microcontroller reduces the continuous sensing energy in two ways.

- The low-power processor's power consumption is similar to that of a typical sensor, waiting during sensor readings does not impose high energy overhead
- Due to the simpler hardware architecture, a low-power processor can transition between sleep and active modes within a very short time.This short transition time allows the sensor processor to be heavily duty cycled to reduce the average power

Usually the main processor of the mobile device is intermittently overloaded with user applications and the operating system, a dedicated processor with sufficient system resources can do real-time sensor pre-processing [1]. Since the dedicated sensor processor can fuse readings from multiple sensors, the proposed architecture also enables richer context aware decision-making through sensor fusion [1].

To implement and evaluate this architecture's efficiency in terms of power consumption and processing capabilities, we use a sensor box which contains accelerometer, gyroscope, magnetometer, barometer, temperature sensor along with ARM based AT91SAM7S256 micro-controller and low power Bluetooth module. Android mobile with OS version 2.3.4 is used as a host to communicate with the sensor box which is obtained from Nokia Research Center(NRC). Sensor box is loaded with the firmware which contains sensor drivers, link drivers, event manager, protocol handler and sensor processing algorithms whereas the Android mobile contains the host communication interface application which send request and receive data to or from the sensor box.

The scope of this thesis is to develop a dedicated sensor fusion and context sensing controller - which could be used to measure any attribute of the surrounding environment and the user, such as the temperature, lighting, noise level, speed or acceleration of the sensor in relation to the environment, and data from different types of sensors can be combined. In addition to pre-processed sensor data, the processor provides rich context information to the device to adapt its performance and user interface more intuitively by adding more value and user experience to the current mobile handsets. In this thesis, only accelerometer sensor is used to extract contextual information.

The important challenge in this task is to create a system framework which supports low power continuous sensing architecture and maximize the recognition accuracy of the classifier while maintaining a low computational load. We believe a reasonable compromise can be achieved between recognition accuracy and computational load by stripping the classification process by combining the different classification algorithms.

The tri-axial accelerometer whose sampling frequency is set as 50 Hz is normally attached to the hip part of the body and leg movements are being analysed. Features such as the Mean, Zero Crossing Rate, Standard Deviation, Peak Frequency are extracted from the pre-processed and noise-reduced linear acceleration signal and finally processed by a classifier.

The proposed feature set is used to classify six different event classes like stand-still, walking, running, cycling, vehicle and sports. In this work, our proposed classifier uses naive-bayesian machine learning algorithm which is computationally easy to implement inside an embedded device like sensor box.

The dataset for this experiment was collected in a supervised study manner with 10 different people performing 6 different physical activity and it is stored in text format. The naive-bayesian classifier algorithm is trained using the dataset and it is implemented in the sensor box using the fixed point C programming language. The algorithm is verified in simulation in Matlab which shows around 92 percent accuracy in classification. Idle activity like stand-still and vehicle suffers from less accuracy of around 80 percent but higher level activities like running, sports are classified with greater accuracy.

2. THEORETICAL BACKGROUND

This chapter discusses context awareness in general, implementations of context awareness in mobile devices, activity-based context awareness and event detection. Research papers on the topics are discussed and results are given when applicable.

2.1 Sensor fusion in brief

The sensor fusion techniques combine the data from multiple sensors like in figure 2.1 which might be either homogeneous or heterogeneous in nature and related information from human input and other information sources to produce more specific high level information and inferences which cannot be achieved with individual or independent sensors. The multi sensory data fusion improves the overall system performance by improved decision making, increased event detection capabilities, lesser false event detection and thus improving the reliability with respect to the sensors information [2], [3], [4].

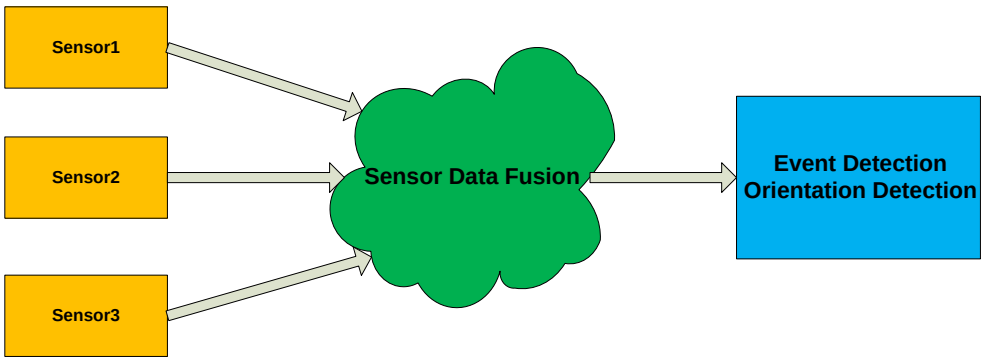


Figure 2.1: Multi Sensor data fusion

In sensor processing the problem due to uncertainty can be eliminated by fusing data from multiple sensors. The amount of fused information and the accuracy of the fused information depends on the quantity and the quality of sensors and the process itself (Sensor data mentioned here could be a raw sensor data or an information/context). Intelligent signal processing and knowledge based systems can be utilized to process the sensor data and perform context sensing and fusion - providing physical contextual intelligence to handsets [5], [6].

2.2 Introduction to sensor processing

The sensor data processing is a process of reconstructing the environment from the sensor data. It is often necessary to process the sensor data to get a more accurate or more complete picture describing the whole situation sufficiently. In sensor data processing, sensor data is taken as input and output is the sensory information, ranging from processed sensor data to recognized events. Modern mobile systems have embraced a variety of sensors, including cameras, microphones, accelerometers, gyroscopes, magnetometer and GPS. Personal-area networking technologies such as Bluetooth provide even more wireless sensors. These sensor signals will then be processed by a measuring device that provides rich information about the physical world, including their human users, or sensory information, to the mobile device. The sensors have fuelled creative developments in what to do with sensory information or sensory applications.

2.3 Inertial sensors

Inertial sensor comprises of an accelerometer and gyroscope which are together used in inertial navigation systems. Accelerometer measures acceleration (rate of change of velocity) while gyroscope measures angular rate. An Inertial Motion Unit (IMU) combines both accelerometer and gyroscope for inertial navigation system.

According to Newton's second law of motion, acceleration of an object directly depends upon the net force acting on the object and inversely proportional to the mass of an object.

$$\mathbf{a} = \mathbf{F}/\mathbf{m} \tag{2.1}$$

Thus the force is equal to mass times the acceleration.

$$\mathbf{F} = \mathbf{m} \ \mathbf{x} \ \mathbf{a} \tag{2.2}$$

Stand-still accelerometer measures 1g upwards relative to the earth's surface while it reads zero g during free-fall conditions. The SI unit of acceleration is metres per second per second $\frac{m}{s^2}$
Gyroscope is the instrument used to measure angular motion. According to Newton, Angular momentum of an object remains unchanged unless it is acted upon by a torque. The fundamental equation describing the behaviour of gyroscope is

$$\tau = \mathrm{d}\, L/\,\mathrm{d}\, t = \mathrm{d}(I\omega)/\,\mathrm{d}\, t = I\alpha \tag{2.3}$$

where the vector τ represents torque, the vector L represents angular momentum,

scalar I represents moment of inertia, the vector ω represents angular velocity and the vector α represent angular acceleration.
A good sensor should satisfy the following requirements.

- Sensor require external power supply for operations so the power consumption is very critical and especially in the case of long term monitoring device
- Sensor should be reliable, robust and durable because the kind of environment of use may vary and also end user may not have sufficient knowledge about its limitation
- Sensor should have self calibration and built in test so that user interaction with it is minimum
- Sensor should have good accuracy and sensitivity to changes

Today's smartphones are often equipped with micro-electromechanical (MEMS) sensors, which have a small form factor and low power consumption advantages without compromising on performance.

2.4 Sensor processing architecture

The current approaches of sensor integration in devices are very rudimentary and would not necessarily enable more intelligent use of sensors. Every sensor is interfaced to the main processor directly and all the sensory data preprocessing and event detection is performed at the expense of the main processor. A dedicated sensor and context processor would improve the power and processing efficiency.

The embedded system should be designed in such a way that system which runs in the microcontroller should not consume more power than that of the microcontroller and sensors and create overhead in data processing.

2.4.1 Event-driven programming approach

An event can be defined as a type of signal indicating that something as happened. As shown in figure 2.2, [10], [14] every major application can be subdivided into two subsection first is event detection and second is the event handling. The event handler is a subroutine which handle event after it is detected. For example, left mouse click triggers an event which calls subroutine i.e event handler to handle the event by opening a new application.

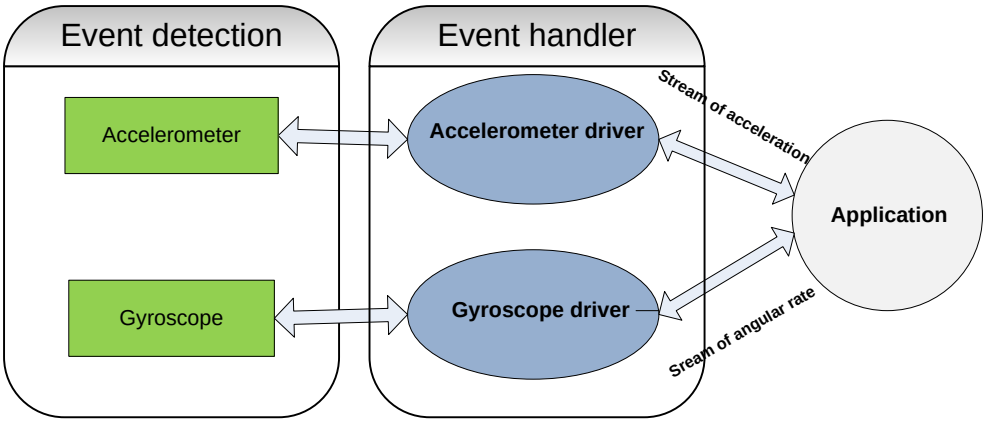


Figure 2.2: Event driven architecture

2.4.2 Source, sink and filter concept

The event-driven programming architecture is based on source, sink and filter concept [48], [49]. Applications sometimes have too much information to process which does not fit into memory and thus are forced to process data in smaller parts. Even when there is enough memory, processing all data may take a long time and thus the application may not be user friendly. Thus complex tasks must be split into smaller tasks which perform series of simpler operation and provides cleaner interface with high degree of generality. These tasks are classified as source, filter and sink.

As shown in figure 2.3, filters can be seen as middle nodes in a chain of data transformations. Sources and sinks are the corresponding end points of these chains. A source is a function that produces data, chunk by chunk and sink is a function that consumes data, chunk by chunk. A filter is a function that processes data received in consecutive function calls, returning partial results chunk by chunk.

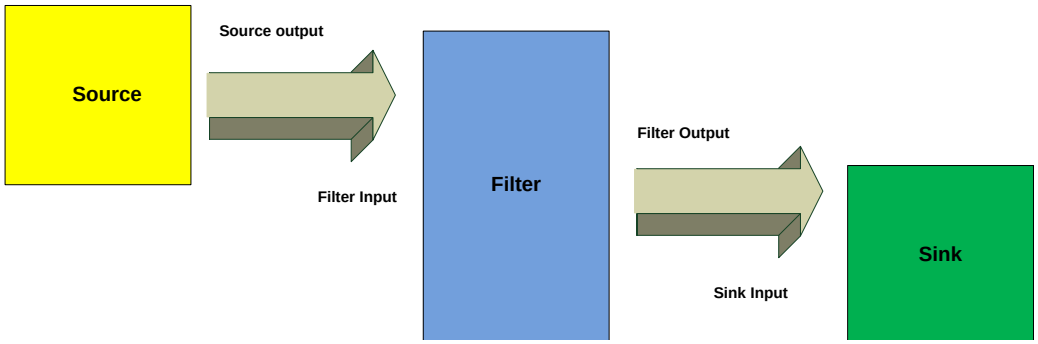


Figure 2.3: Source, sink and filter

2.4.3 Continuous sensing

Nowadays, mobile devices are becoming integral part of the human activities so the need for developing continuous sensing application is growing day by day as described in [11], [12]. Continuous sensing may enable large number of use cases related to context recognition and activity monitoring. Mobile devices could also appear much more responsive than previously. Wireless sensor navigation, augmented reality and many other applications can benefit from continuous sensing.

Continuous sensing based application continuously samples the sensor data and hence the power consumption of this application will drain the battery in a short span of time. Continuous sampling of sensor data prevents the mobile device's application (host) processor from entering the low-power sleep state thereby increasing its power consumption. The need to keep the device's main processor and associated high-power components active to access the sensor results in a total power consumption that is larger than the power required for actual sensor. According to [11], [19], [13], the energy efficient continuous sensing application can be achieved by a good system architecture. As discussed in [13], the whole system architecture can be implemented in two different ways basically with and without dedicated microcontroller

As shown in figure 2.4, the sensor system implemented inside the host processor is running at the expense of the host processor power which is very expensive. In this case, the host processor cannot go into the sleep mode often.

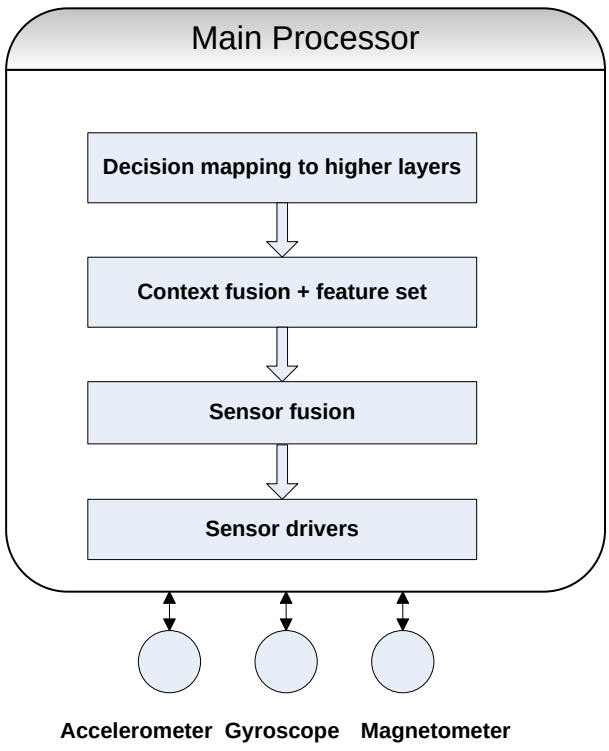


Figure 2.4: Sensor architecture without dedicated microcontroller

In figure 2.5, the sensor system is implemented inside a dedicated microcontroller and the subsystem is running inside a low power microcontroller which has much lower power consumption. Host communication interface connects the microcontroller and the host processor which can be I2C, Bluetooth, USB, etc.
The main differences between these two architectures are as follows:

- The microcontroller firmware contains the driver for peripherals, clock rate for the MCU, queue scheduler etc, these are not needed when algorithm runs in the host processor
- Due to the standalone microcontroller, host processor can get into the sleep state very frequently in continuous sensing scenarios
- The overhead due to host communication interface and microcontroller firmware are negligible when running the entire algorithm in the standalone microcontroller

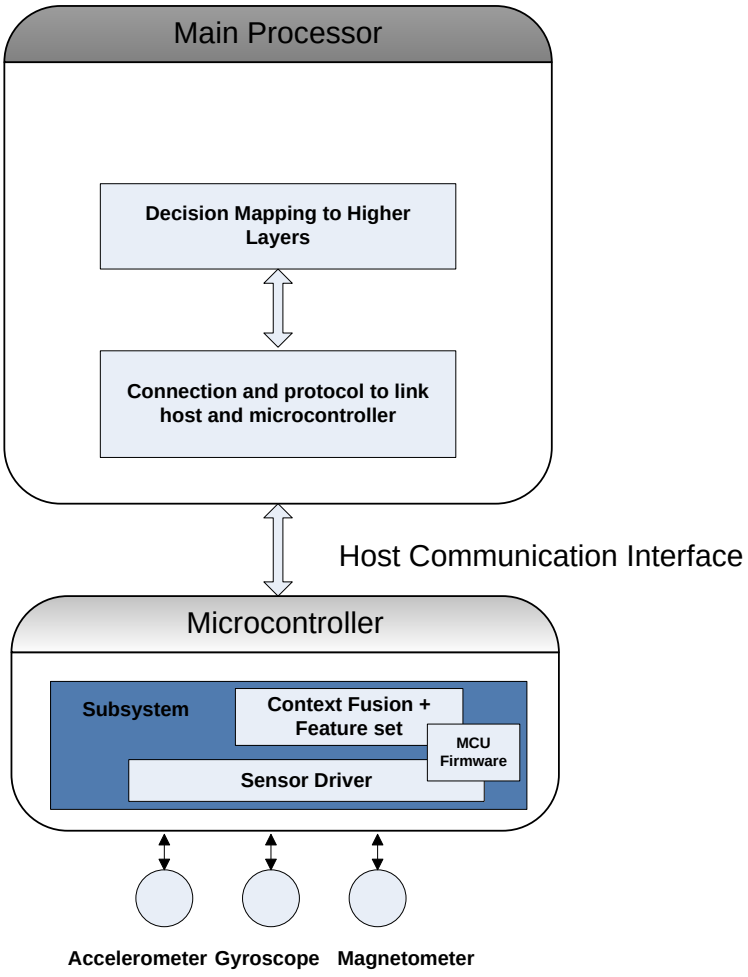


Figure 2.5: Sensor architecture with dedicated microcontroller

2.5 Context awareness

Context is any information that can be used to predict the situation of an entity. The entity might be both the user and the device in an environment. Context awareness relates to a device's ability to be aware of its environment, user action and its own state and adapt its behaviour based on the situation. There is no unified theory to classify the contexts a device can sense. Comprehensive set of context classes discussed by [16], [17], [18] which the device can sense are as follows:

- **Environmental Context** classes define the user's surrounding environment example, noise, sound, temperature, light, altitude etc
- **Activity Context** classes defines what the user is doing, for example person driving in a car, cycling, sitting, walking etc
- **Physiological Context** classes defines persons heart rate, blood pressure etc
- **Mental Context** classes defines person's mood and stress level
- **Spatio-Temporal Context** classes refers to aspects of space and time example, direction, location, speed etc
- **Terminal Context** classes defines end terminal equipment which the user is interacting example, Bluetooth, WLAN etc

The paper [15] discusses about context classes which could be extracted from various sensors attached to the mobile handsets. The Table 2.1 provides the context information extracted from sensors.

Sensor	**Context Information**
Accelerometer	**User Activity** - Standing, Sitting, Walking, Running, cycling
	Orientation - Face Up/Down, Portrait/Landscape
	State - Idle, Motion
Gyroscope	**Orientation** - Face Up/Down, Portrait/Landscape (w.r.t Earth)
Magnetometer	points to Magnetic North
	Tilt compensation to Accelerometer

Table 2.1: Context Information associated with Sensors

The available context information can thus be tailored for specific instance of the device. After sensing the situation the device can adapt its functionality at run time.

This continuous sensing and context awareness creates many use cases in the field of mobile communication.

- Combination of pedometer and GPS combined with the wireless cell ID can track the time and movement of the user while in office, home or walking
- An accelerometer or any other sensor based algorithm runs continuously in the background. When user makes a specific gesture, e.g. picks up the phone, pinch the phone, a dedicated processing sequence can be initiated
- With the help of microphone signal running continuously in the background, with user's sound, the phone detects makes an action e.g. a voice command
- Continuous context sensing can also save power by switching off irrelevant functionality such as WLAN, GPS when the mobile is idle
- It can do some location based services e.g. when the user is in office mobile can switch to WLAN instead of using the internet

2.6 Physical activity recognition

Automatic recognition of physical activity of human is one of the important and challenging areas of ubiquitous computing. Physical activity recognition models are not only used in mobile devices but also in health care monitoring purposes [24]. The current generation smart phones contains tri-axial accelerometer which can be used for detecting the physical activity of the user such as stand-still, walking, running, cycling, vehicle and sports. With the help of tri-axial accelerometer, the magnitude of the acceleration along each axis can be found which includes the vector sum of both gravitational and linear movements. The sensor is placed carefully in the circuit board without any loose connection to avoid artefacts being produced during measurements which can also be eliminated by signal filtering.

The various applications of accelerometer are shown in table 2.2. Waist is the best location to detect the physical activity of humans because center of gravity is concentrated in that area. The frequency range of various human activities as described in [30] ranges from 0.3 to 3.5 Hz. The dynamic range required for the measuring various human activities depends upon the application. Running and sports requires more dynamic range than cycling and walking. Most of the modern accelerometer has the capability to measure upto $\pm 16g$ which can be used in mobile devices for various applications.

The usage of multi sensory or single sensory approach for detecting the activity depends upon the application but in general multi sensor approach gives better classification result than using single sensor which is described in detail in [31] and [7].

There are many useful applications that can be built around accelerometer for example, the orientation of the device (landscape or portrait mode) can be found

Location	Application
Ankle and thigh	Leg movement during walking [25]
Wrist	Daily living activities [26]
Waist	Detection of activity [27]
Multiple location	Behavioural research [28]

Table 2.2: Accelerometer location and application

with the help of gravity vector. In the accelerometer based physical activity recognition module, activity report can be generated on weekly, monthly basis which can be emailed to the user. This activity report can help user to correct his health practices and can also alert him if he is not getting adequate amount of exercise. This activity information can be also used to adapt the behaviour of the mobile such as music could be loaded automatically to match the user activity.

2.7 Free-fall detection

Free-fall detection is important in consumer devices because the algorithm helps to take preventive measure before the device makes an impact with the ground. Thus free-fall detection helps device to secure the drive and its data and also helps to put the device in the safe mode to minimize the damage. Accelerometer, an inertial sensor can be used to detect free-fall events in hand-held devices.

Accelerometer measures linear acceleration(A) by subtracting the gravitational vector(g) from the movement acceleration(a) along the input axis.

$$A = a - g \tag{2.4}$$

During free-fall condition, $\mathbf{a} = \mathbf{g}$ which means tri-axial accelerometer indicates zero values along all the three axes simultaneously. In this case, application simply needs to monitor for the zero-gravity and as soon as it detects the zero-gravity it needs to alert the main processor. Accuracy of the algorithm needs to be good as there can be false detection of free-fall situation and also it clearly needs to distinguish user activity such as jumping, running from actual free-fall situation [35], [36].

2.8 Classifiers - overview

Classifiers helps in recognizing and classifying the input feature set to specific user defined class labels. Feature extraction is the process used to extract the key elements from a processed signal which made the signal distinct which are used by the learning algorithm to produce the decision threshold.
There are two types of classifiers as described in [29]

- **Supervised classifiers** are trained with the large training data with the help

of learning algorithms

- **Unsupervised classifiers** takes input training data and searches for the clusters without knowing the class labels associated with the data

As shown in figure 2.6, classifier take the training data to extract the feature set and define decision thresholds using the machine learning algorithm and then classify the real test data depending on the decision threshold obtained from learning.

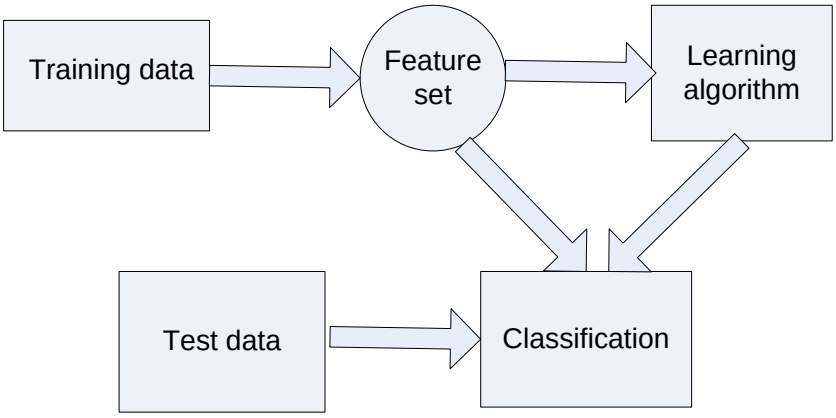


Figure 2.6: Classification steps

Learning from the training data is a computationally complex functionality which is usually done off-line e.g. in computer and the classification itself is done inside an embedded device. The classifiers are the same as those generally used in machine learning and pattern recognition. Supervised methods are generally preferred, because they find exact matches with real-world classes. Classifiers are choosed based on classification accuracy, memory requirement and computational complexity. Generally classifiers with low memory and low computational complexity is chosen. There is always a trade-off between memory/computational load vs accuracy.

Several algorithms are available for taking decision on different states and the key advantages and disadvantages between these algorithms are discussed in papers [20], [21].

1. **Naive Bayesian classifier**:
   Naive bayesian classifier uses normal distribution to calculate the probabilities as given in [39], [44], [45].

$$[P(X) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(X-\mu)^2}{2\sigma^2}) \tag{2.5}$$

   where μ is the mean of the signal X and σ is the standard deviation of the signal X and σ^2 is the variance of the signal X.

Advantages: Learning is very fast, easy and robust. Calculating probability is very straight forward with normal distribution
Disadvantage: Considers that all parameters are independent this is why it is called "naive"

2. **Decision trees**
Decision trees classify instances by starting at the root of the tree and moving through the nodes until it reaches the leaf node

Decision trees represents the following:

- Each internal node present in the decision tree represent an attribute
- Each branch in the decision tree corresponds to attribute value
- Each leaf node represent the classification value

The construction of decision tree can be done as follows:

- Selection and decision on node terminal and splits from the parental node
- Assignment of each class to the terminal leaf node

Decision trees can be constructed with the help of ID3 algorithm [40], [41] and C4.5 algorithm [42], [43] or can be customized for particular set of instances.

Advantages: Classification is robust and depends upon the threshold used in making decision. Ability to work with both continuous and discrete value attributes. Decision trees can also work with noisy training data
Disadvantages: Decision trees takes a longer time to create. It needs separate methods to specify tree height and also possible over-fitting

3. **Multilayer perceptron**
As shown in figure 3.2, multilayer perceptron uses neuron which is the artificial processing unit. A multilayer perceptron consists of several numbers of neurons and processing of only one neuron is shown in figure 3.2. Input signals are multiplied with the individual weights which are then added up and fed to the activation function as described in [46], [47].

Advantages: Classification accuracy are very precise with good training data. It requires no details about the system
Disadvantages: Choosing number of neurons in the hidden layer is a problem due to over training or bad classification. Training the neural network can be very time consuming

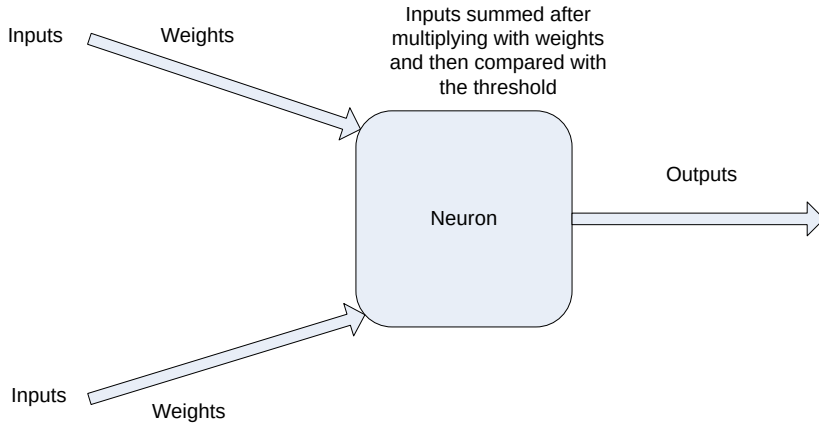


Figure 2.7: Multilayer perceptron model

Naive bayesian classifier is choosen for an implementation as it is the simplest and robust method for the implementation in embedded device. For effective classification it is important to choose features which can discriminate the classes. The features which are selected for naive bayesian classifier are Mean, Variance, Standard Deviation, Zero Crossing Rate and Peak Frequency component of the signal. The emphasis is on computationally less complex feature that can be computed in the embedded device.

1. **Mean**: The Mean μ represents mean of values within the sliding window which is represented in equation below. This feature is calculated on the accelerometer input data which helps to distinguish horizontal and vertical postures of humans.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x[i] \tag{2.6}$$

   where n denotes the window length and x[i] denotes the i^{th} sample

2. **Variance**: The Variance σ^2 represents the variance of the signal around the sample mean within the sliding window. It represents the signal energy

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x[i]-\mu)^2 \tag{2.7}$$

3. **Zero crossing rate**: The Zero crossing rate (ZCR) generally represents the average amount of time the sign of the signal changes from positive to negative or back. ZCR also represent the mean crossing rate where average number of time the sign of the signal changes around the mean value. Mean crossing rate

is calculated on the accelerometer input which helps to distinguish the type of human motion

4. **Peak frequency**: The frequency spectrum of the signal x can be found by using Discrete Fourier Transform (DFT) explained in [37]

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \tag{2.8}$$

where X represents the frequency spectrum, n represents the index and N represents the size of the data.
Fast Fourier Transform (FFT) is used to calculate the Discrete Fourier Transform (DFT) of the signal efficiently by using the cooley tukey algorithm [38]. The Power Spectral Density (PSD) of the signal is obtained by squaring the the discrete time frequency component of the signal

$$P(f) = \frac{1}{N} |X(f)|^2 \tag{2.9}$$

Thus peak frequency is the frequency which has the maximum computed power spectrum. The dominant frequency of the accelerometer data can be used to distinguish running from walking. The frequency of running is higher than that of walking due to higher step rate during running.

3. RESEARCH METHODS AND MATERIAL

This chapter discusses the implementation of physical activity detection in a dedicated microcontroller which is connected to the Android mobile through Bluetooth interface. The design of system architecture and the implementation of firmware modules are discussed in a detailed manner with the help of figures and flowcharts.

3.1 System architecture overview

The system architecture of the sensor box shown in figure 3.1 contains both hardware which is explained in section 3.1.1 and firmware portions which is explained in section 3.1.2. The Bluetooth communication interface which connects the mobile device and sensor box is explained in section 3.1.4.

Main processor acts as a master which initiates the processing of sensor processing module present in the sensor box and thus the sensor box acts as slave. The sensor subsystem is implemented inside a dedicated micro-controller present in the sensor box which has a very low power consumption. Host communication interface is an application installed in the mobile devices which controls the sensor box through Bluetooth interface. The advantage of this system architecture is that the signal processing algorithm is implemented in a standalone low power microcontroller and hence the main processor can go to the sleep state very often and hence reducing the power consumption.

The hardware and software initialization are done by the sensor subsystem when it is switched on and finally waits for the command from the main processor through the Bluetooth interface.
The hardware initialization are as follows:

- Clock is configured in the microcontroller
- General Purpose Input/Output (GPIO) pins are enabled
- Interrupts are disabled
- Universal synchronous/asynchronous receiver/transmitter (USART) is initialized
- RS-232 is initialized

USART and RS-232 are helpful for debugging the sensor box which can be connected to the laptop for checking the debugging prints in the serial console.
The software initialization includes the following activities:

- Memory initialization describing the start and end of memory allocation and also the heap size in the microcontroller
- General Purpose Input/Output (GPIO) pins are initialized
- Two Wire Interface (TWI) slave bus is initialized
- LED initialization and setting to ON state
- Flash initialization
- Event manager initialization
- Protocol manager is initialized and it creates Bluetooth link between the sensor box and mobile device

After initialization, sensor subsystem monitors link for the input from the main processor. When protocol manager register a command from the main processor through Bluetooth link, it starts the module and event manager so that the corresponding module in the sensor subsystem is started and finally event manager registers the data as shown in figure 3.2.

The software is implemented in C language and the program code size for various modules are given below in table 3.1. Drivers takes approximately 2 KB and firmware portion takes 4.5 KB program code size approximately. free-fall detection module takes 0.5 KB approximately and physical activity module takes 14 KB approximately.

Module	**Code size(bytes)**
Accel driver	850
Mag driver	500
gyro driver	750
event manager	1750
protocol	3200
free-fall	450
physical activity	14200

Table 3.1: Program code size

3.1.1 Hardware architecture

The sensorbox obtained from Nokia Research Center (NRC) contains the following components:

- Atmel Microcontroller at91sam256ek
- USB controller
- Accelerometer - LIS331DLH
- Gyroscope -
- Magnetometer -
- Barometer - VTI SCP-1000
- Temperature sensor
- Bluetooth chipset -

The sensors are connected to the micro-controller through TWI/I2C slave. It also has one JTAG port for programming and debugging. The sensor box has a built in battery support which can be charged through standard Nokia charger.

3.1.2 Firmware architecture

The sensor subsystem which is implemented inside the sensor box as shown in figure 3.1 is essentially a framework for sensor drivers and algorithms. The design is build around modules that implement module interface. Sensor drivers acts as module that outputs sensor data and accept commands from main processor as their input.

As described in figure 3.1 mobile devices acts as a master and sensor box acts as a slave. Data communication between the main processor and sensor box are handled as events as shown in figure 3.2. This includes commands from host system and also events that are produced by modules such as sensor drivers or sensor processing algorithm as shown in figure 3.2. Modules are usually sensor drivers or signal processing algorithms such as context sensing.

As shown in figure 3.2 accelerometer driver outputs sensor data that are handled as events by event manager which are consumed by the main processor through link drivers. Algorithms are similar except they process input sensor data and produces output events. Protocol is another module that accepts all event types as input including commands from host processor.

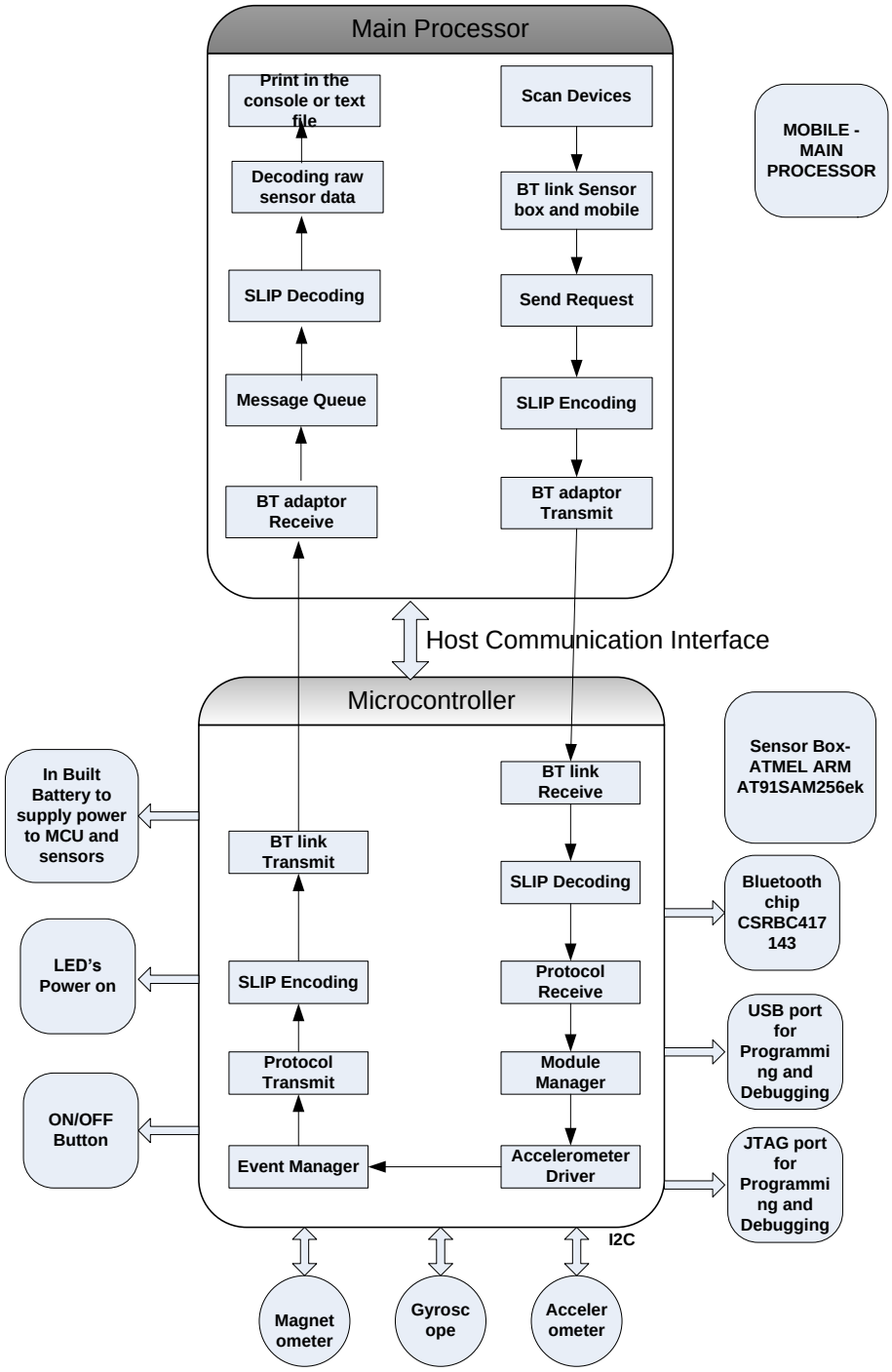


Figure 3.1: System architecture

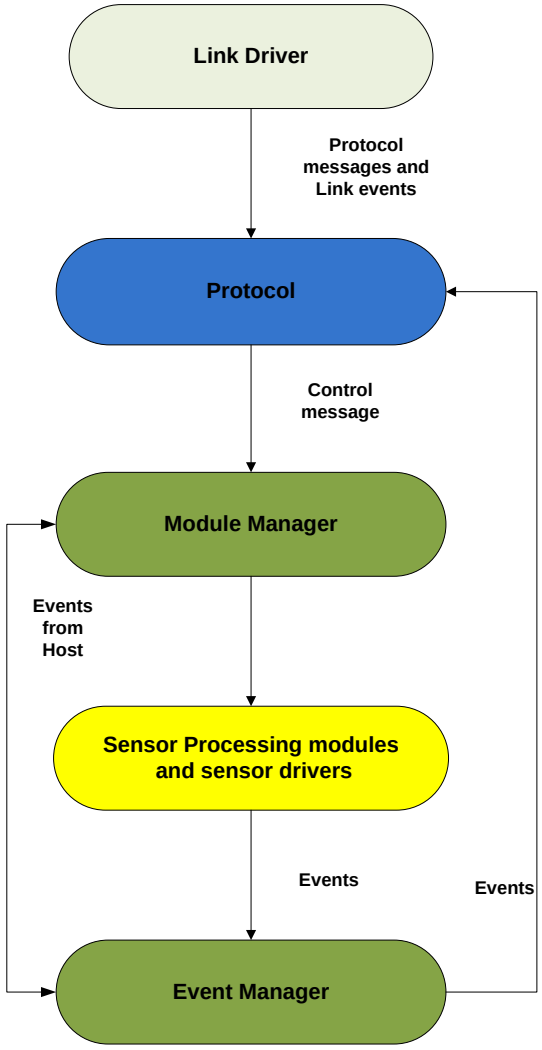


Figure 3.2: Sensor subsystem overview

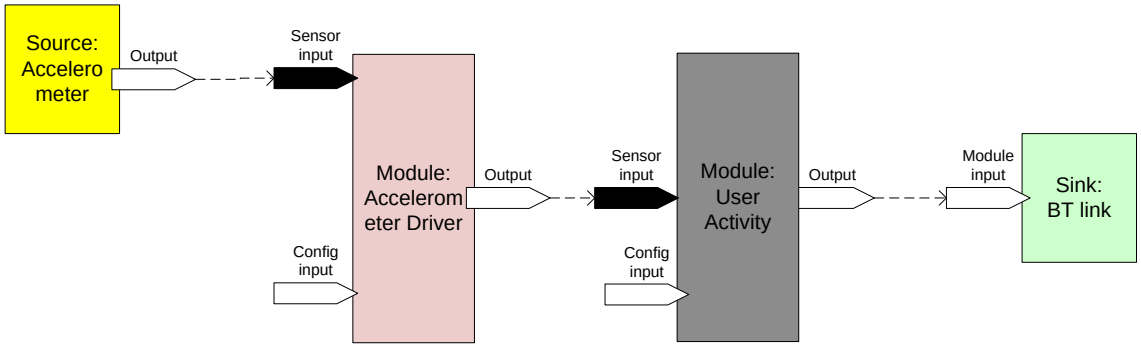


Figure 3.3: Physical activity module - source, filter and sink classification

The microcontroller has limited program and data memory, hence cannot be used to store and process real time sensor data. The source, sink and filter concept as discussed in section 2.4.2 is implemented to process the sensor data. As shown in figure 3.3, Sensors are sources producing data while protocol and Bluetooth link

which connects the host processor are sinks. Between sources and sinks lies the module interface such as accelerometer driver and physical activity module.
Module implements filter which receive input from source and passes output to the sink. Filters produce data only when they receive input from sources. All sensor and context algorithms are considered as filters. Since algorithms often have an internal state it becomes necessary to have several instances of the same algorithm.
There are two types of inputs: plain inputs and inputs that trigger processing.

Plain inputs are essentially variables that hold the last value written by the sources. Both the event manager and the module owning the inputs can directly access them as shown in figure 3.2. For example: Accelerometer (Sensor box) —> Host (Main processor) The events produced from the accelerometer sensor is transferred to the main processor. In this case, the main processor can access the accelerometer data.

Trigger inputs are used for implementing data processing as shown in figure 3.3. when an event is delivered to input of module which does signal processing algorithms, using the same mechanism as for plain inputs, the event processing function of the module interface is called. For example: Accelerometer —> Physical Activity and Physical Activity —> Host.

In this case, accelerometer is started and the output of the accelerometer is processed by physical activity module. Thus the final output of the physical activity module is passed on to the main processor. Here, the host processor displays only the context information as shown in table 3.5.

Each module can have any number of output items. Each output item is associated with a data type, data format and size. Outputs can be connected to compatible inputs. The definition of data type is different from most of programming languages. Data type defines the semantics of data which gives meaning to it. For example, "acceleration" or "activity", the format defines how the data is stored in the memory. Format can be variable or structure. For example, formats include 32-bit signed integer or 3-dimensional 16-bit signed integer vector.

Considering accelerometer values are typically ADC values with dynamic range represented in G, so a 12 bit ADC resolution is sufficient for 1G sensor which means then 2047 would represent 1G acceleration and -2048 would represent -1G acceleration. Conversions from raw values typically only require bit shifts.

3.1.3 Android Bluetooth architecture

The Android Bluetooth class in [22], [23] provides functionality to scan, connect and manage data connection transfer between bluetooth devices.

The steps for establishing the Bluetooth connection between the Android device

and sensor box includes scan for the Bluetooth device, pairing the devices, establishing the RFCOMM channel connection between the device and the sensor box and finally managing the data transfer.

The figure 3.4 explains connection between various Bluetooth devices with that of Android mobile. For connecting mobile and sensor box, RFCOMM socket connection needs to be established between them before transmitting the data.

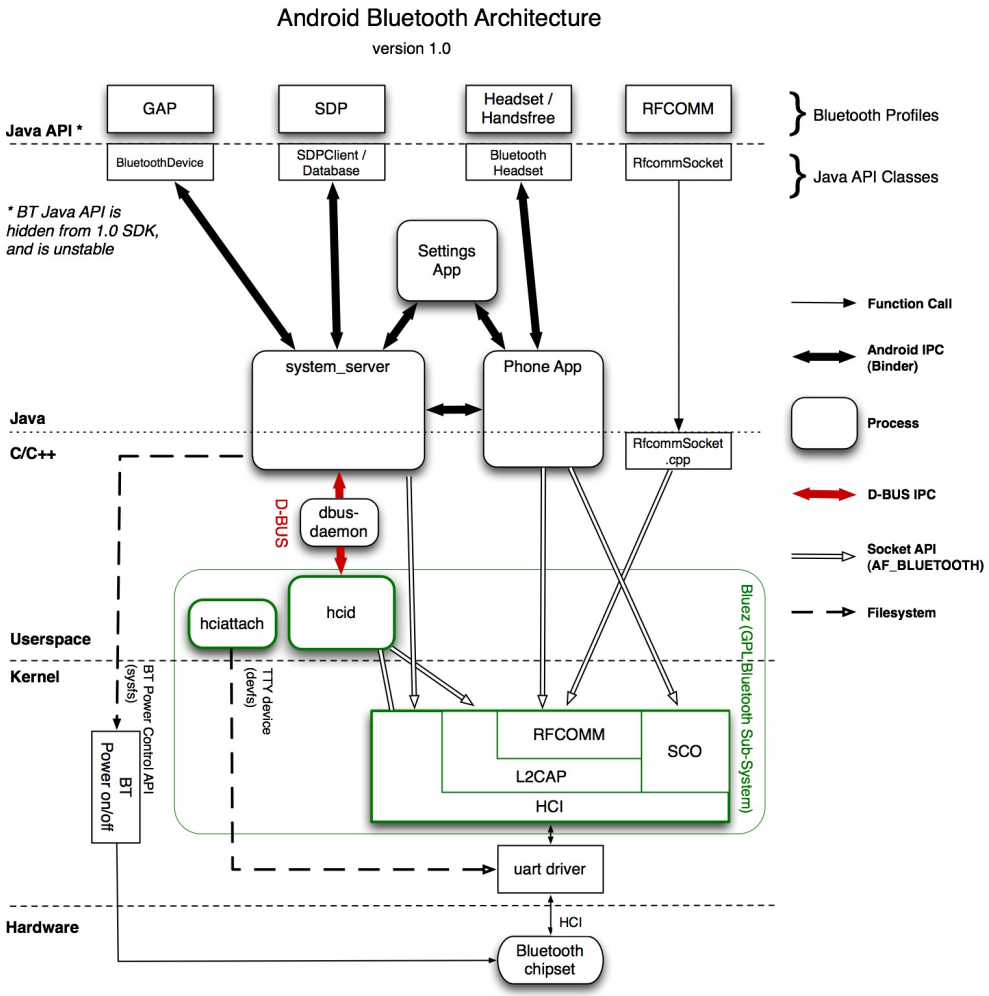


Figure 3.4: Android Bluetooth architecture

3.1.4 Host communication interface

The Bluetooth communication interface as shown in figure 3.1 links the sensor box and Android mobile. It is implemented in the mobile in the form of application using Android Software Development Kit (SDK).

The Android application contains both the Bluetooth adaptor and the communication interfaces functionality. The Bluetooth adaptor is implemented with the help of Android SDK in Java. The communication interface functionality of Android

application is implemented in C language with the help of Android Native Development Kit (NDK). NDK allows to implement part of the Android application in native code such as C, C++ using JNI interface [32]. This interactive application which runs in Android mobile is shown in figure 3.5. Threads are implemented using Android thread class [33] to speed up the sensor data processing.

The Android host interface application as described in figure 3.5 implements the following functionality:

- The Bluetooth is switched on manually in the mobile
- The application has a scan devices button if pressed/touched searches for the nearby Bluetooth devices and list the device name and MAC address. This procedure is knows as discovery mode which is explained in [22]
- If the target sensor box is selected from the list, the application stores its MAC address in its memory. This is known as pairing. Then socket connection is established between the mobile device and sensor box. RFCOMM channel provides a secure encrypted connection for transmitting and receiving data over the air
- Data transfer is initiated by a byte command, sent from the application to start the target sensor present in the sensor box is shown in table 3.3 and table 3.4. This command is encoded with the help of Serial Line Internet Protocol (SLIP) encoding mechanism before transmitting via Bluetooth which is explained in figure 3.6
- The received sensor data from the sensor box is pushed to the message queue continuously and a separate thread is started to read the data from the message queue
- The received sensor raw data needs to be decoded by the same SLIP mechanism to get the original sensor data. Each raw data from the message queue is then decoded as shown in table 3.2

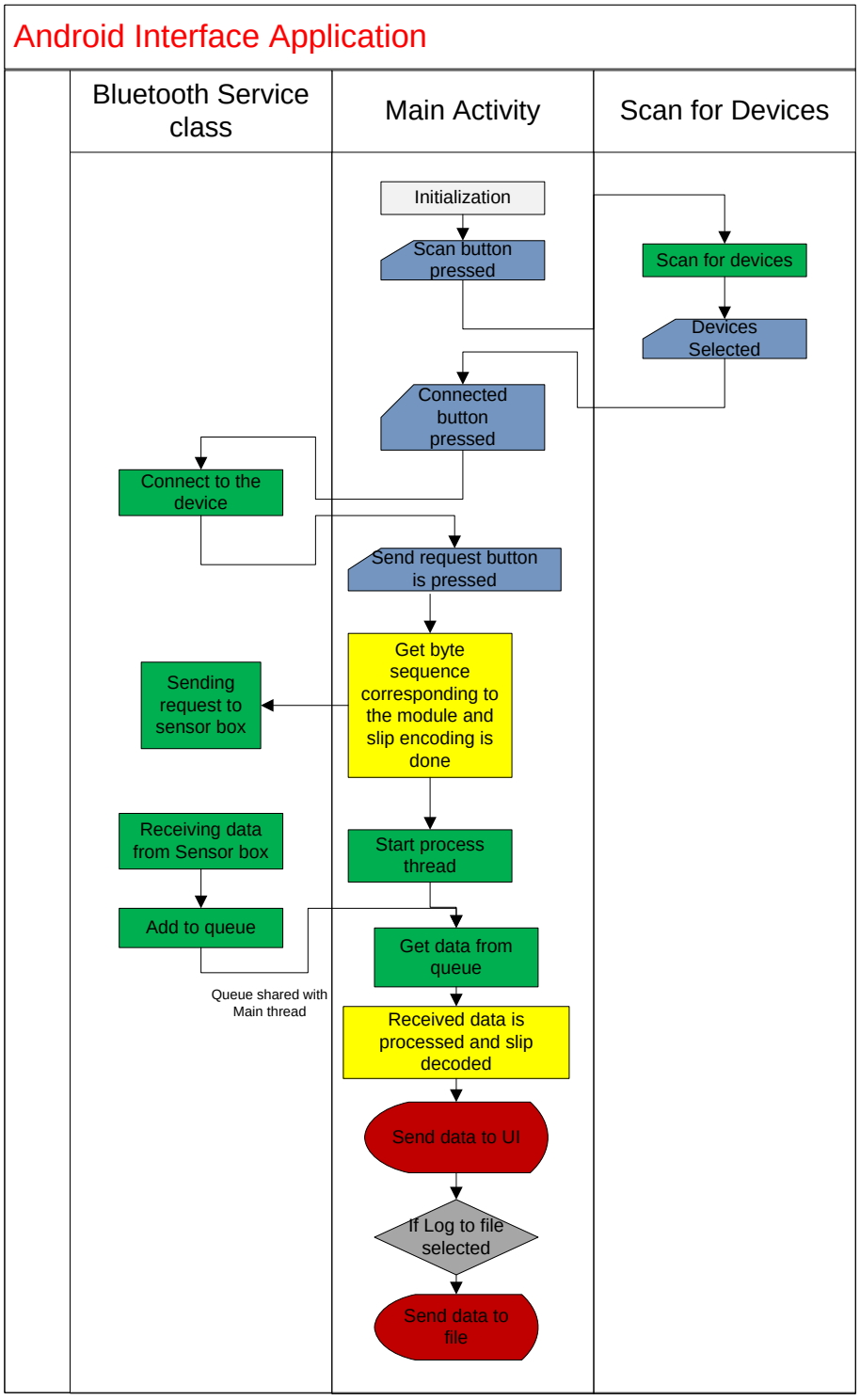


Figure 3.5: Android host interface application

The Bluetooth events coming from the sensor box are registered with the message queue and a separate thread is started to decode the information from the raw sensor data which is shown in figure 3.5. The message queue is important because without it, application present in the main processor misses other Bluetooth events. The decoding of raw sample is done after the SLIP decoding procedure. Each 13 byte

of raw sensor data as described in table 3.2 contains few bytes to describe the data format, data type, payload data size, link id, time stamp information and finally the payload.

0	1	2	3	4-5	6-12
Data Format	**Data Type**	**Data size**	**Link ID**	**Time stamp**	**payload**

Table 3.2: Raw data decoding information

The Serial Line Internet Protocol (SLIP) is a character oriented protocol to send packets over the air [34]. This character oriented protocol identifies the beginning and end of the frame using a special character. SLIP defines two characters END (0xC0) and ESC (0xDB) which are in hexadecimal notation are stuffed in the frame to identify the start and end of the frame.

SLIP works as shown in figure 3.6

- Data are encoded with SLIP before transmitting
- If a data byte is same as END, a 2-byte sequence of ESC and 0xDC is sent instead
- If a data byte is same as ESC, a 2-byte sequence of ESC and 0xDD is sent instead
- END is transmitted both at the start and end of the frame

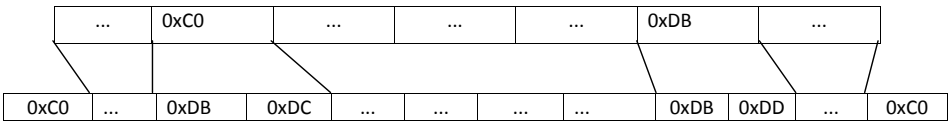


Figure 3.6: SLIP Encoding

The module information are shared between the subsystem present in the sensor box and the Android application. So the configuration and input/output parameters are stored in a linked list which is explained in table 3.3. When an accelerometer "start" command needs to be sent from the application in the host processor, it searches for the corresponding node and retrieves the information in the form of data

Module Name	Name of the module.
Module type	Source, sink or filter
Real time Flag	Flag which tells whether processing is real time or not
Sensor/Context Flag	Flag which distinguishes sensor from context processing
Data Type	data type of the module.

Table 3.3: Information in the Linkedlist

Source	**Sink**	**Filter**	**Start/stop Flag**	**Data type**

Table 3.4: Module information

packet as described in table 3.4 which are then SLIP encoded before transmitting via Bluetooth.

A separate node is created in the linked list with information described in table 3.4 for each module. The information from the linked list are printed into a separate header files and it is copied manually to the sensor box. This header files are very important to successfully compile the subsystem present in the sensor box otherwise subsystem is not aware about its modules. This method is followed to link all sensor processing modules present in the subsystem and whose informations are also known to the host application. There should be an one to one relationship about the module information between the application and the subsystem which is done through the header files.

When the sensor box retrieves the 5 byte array message as mentioned in table 3.4, it decodes the messages as explained in figure 3.1 and figure 3.2, the information already stored in the header file are used to start the correct module.

3.2 Sensor processing modules

3.2.1 Physical activity context classification

The physical activity recognition algorithm takes input from accelerometer sensor data and classify signals into six classes explained in table 3.5. Machine learning technique is applied on the sensory data to classify the activity of the user as mentioned in table 3.5.

Class No	Class Name
1	Stand-Still - Standing, Sitting
2	Walking
3	Running
4	Bi-Cycling
5	Vehicle
6	Sports

Table 3.5: Physical activity class information

The purpose of this experiment is to analyse the accuracy, stability and finally implement the naive-bayesian classifier in the subsystem present in the sensor box. For this experiment, tri-axial accelerometer sensor present in the sensor box is used and the signal is sampled at 50 Hz with 16 bits then a first order low pass filtering is done.

Tri-axial accelerometer is widely used in activity detection application because it responds well to both acceleration due to gravity and acceleration due to body movements, thus making it reliable to estimate both the postural orientation as well as body movements. The sensor box is placed in the pant pocket (Waist location) because the activity of the user can be detected accurately from the movement of the legs as explained in section 2.6. Only Linear acceleration is used in sensing physical activity and thus the gravity vector is removed.

Linear acceleration = acceleration - gravity vector

There are two stages in this context classifier, first is the learning stage and second is the detection stage. Database containing accelerometer samples are created with 10 people performing all the activities such as stand-still, walking, running, cycling, vehicle and sports.In supervised learning stage as shown in figure 3.7, signals of all users from each class are read in Matlab, then the three axes are merged using vector length measure ($\sqrt{x^2 + y^2 + z^2}$) and finally filtered using first order low pass filter. The next step is to calculate the feature vectors from the linear acceleration signal for each class separately such as Zero Crossing Rate, Mean, Standard Deviation, Peak Frequency of the signal. These feature vector values for each of the six classes are stored in a lookup table which is used in detection.

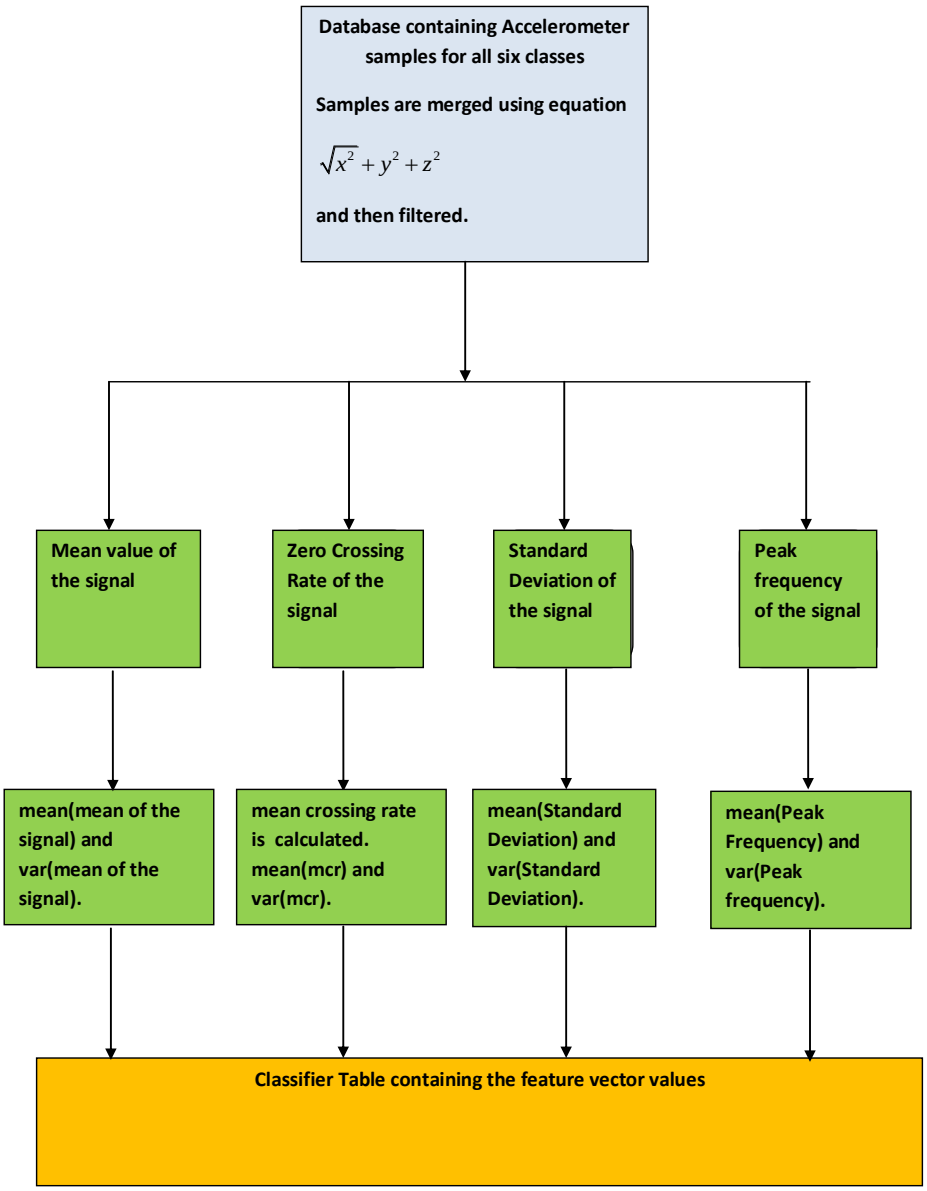


Figure 3.7: Learning stage

The classifier table contains mean and variance of feature vector values for each of the six classes as required by the naive-bayesian classification model which is shown in table 3.6. Thus classifier table is a 8×8 matrix which stores the feature vector values as shown below in figure 3.7 and it is used in the detection stage.

In the detection stage as shown in figure 3.8, a real time tri-axial accelerometer signal of 512 samples are fed through the classifier which calculates the feature vectors such as Mean Crossing Rate, Standard Deviation, Peak Frequency and Mean value of the signal. Then the naive-bayesian classifier model which uses the normal

No	Parameters in classifier table
1	Mean(Mean crossing rate)
2	Mean(Standard deviation)
3	Mean(Peak frequency)
4	Mean(Mean value)
5	Variance(Mean crossing rate)
6	Variance(Standard deviation)
7	Variance(Peak frequency)
8	Variance(Mean value)

Table 3.6: Classifier table

distribution to estimate the probability is calculated as follows.

$$P(X) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(X-\mu)^2}{2\sigma^2}) \tag{3.1}$$

where μ is the mean of Mean Crossing Rate, Standard Deviation, Peak Frequency and Mean value from the classifier table, σ^2 is the variance of Mean Crossing Rate, Standard Deviation, Peak Frequency and Mean value from the classifier table. X represent the values of the feature vector calculated during the detection stage.

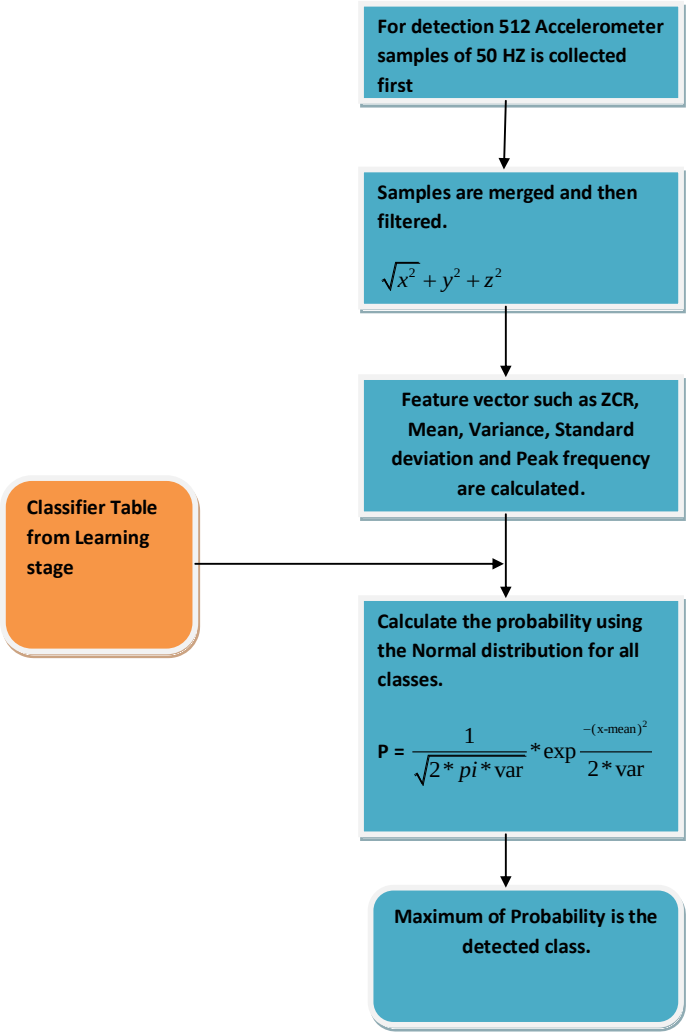


Figure 3.8: Detection stage

Thus probabilities are calculated for each of the four feature vector and finally multiplied to get the final probability of the class. The detected class corresponds to the maximum probability value.

The physical activity module has two parts, one is the sample collection and the other is the context algorithm. So depending on method of collecting samples, the module can be scheduled by the subsystem present in the sensor box in different ways as explained below.

In figure 3.9, the algorithm is scheduled in a simpler and linear manner in which a subsystem waits till it gets the required amount of samples before scheduling the

activity context. In this case, fresh sensor data are collected every time.

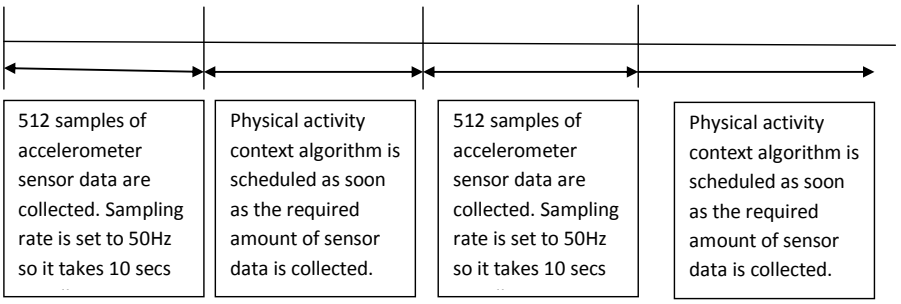


Figure 3.9: Physical activity algorithm - scheduling method 1

In figure 3.10, activity context algorithm is scheduled similar to figure 3.9 but the only difference is that samples of previous iterations are stored and it is used in next iterations thus saving half of the time for collecting samples. Since this method depends on the previous samples the real time classification accuracy has some latency.

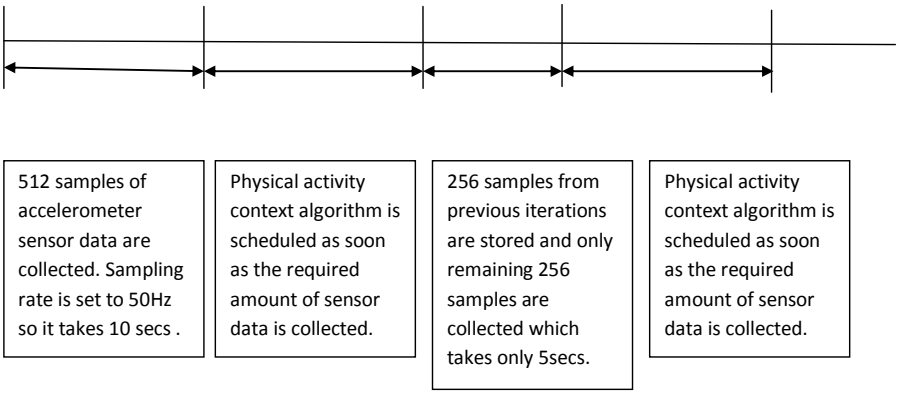


Figure 3.10: Physical activity algorithm - scheduling method 2

In figure 3.11, samples are collected for the next iteration in parallel to the algorithm which requires task scheduler in the embedded subsystem to manage these tasks. Also, the classification accuracy has more latency when compared to the previous two methods.

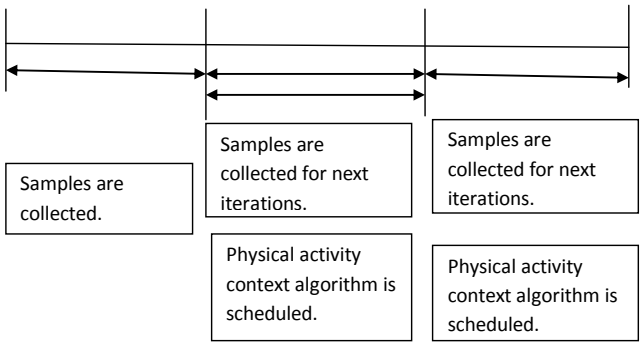


Figure 3.11: Physical activity algorithm - scheduling method 3

Implementation method based on figure 3.9 is choosen for this experiment as it is simpler to schedule and test the validity of the context algorithm in real time.

Microcontroller is programmed such that it runs at 30MHz with 100 percent duty cycle which means microcontroller is not entering the sleep mode because of the active bluetooth link between the host and the sensor box. So with the help the datasheet of the microcontroller at91sam7s256 [50] the active current consumption is found at around 20mA which is less than than that of ARM cores present in the smart phones.

3.2.2 Free-fall event detection

The free-fall event detection module is implemented inside the subsystem and this module monitors the input accelerometer sensor data and outputs true when free-fall is detected. Simple threshold based free-fall detection method is choosen for implementation.

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3.2}$$

if $(a > a_{th})$ then no free-fall alarm.
if $(a <= a_{th})$ then free-fall alarm is raised.
whereas $(a_{th} = 0.5g)$ is the threshold set inside the module, and (a_x, a_y, a_z) are x, y, z components of acceleration.

There are some irregular user activities which are not counted as free-fall events. Examples like drop with a spin and toss and catch, because of their non-repetitive nature, are difficult to distinguish from the free-fall event detection.

4. RESULTS AND DISCUSSION

4.1 Testing the sensor subsystem

The sensor box contains firmware which includes the hardware/software initialization, sensor driver, protocol, event manager and sensor processing algorithms. This subsystem is tested with the help of test framework which resides in the application program.

The main objective of the test framework is to test the sensor drivers, sensor data and context algorithms in the sensor box. Testing means querying the sensor box through Android interface program present in the host processor for a specified amount of time with appropriate input parameters and log the response from the sensor box to a file as shown in figure 4.1. Testing the sensor is very important because of the fact that the sensor data are used both during learning and also in detection stage. Any loss in sensor data will result in loss of information and eventually affect the accuracy of the classification.

Test functionality does not include any drivers or third party software to tests the functionality of the overall system framework.

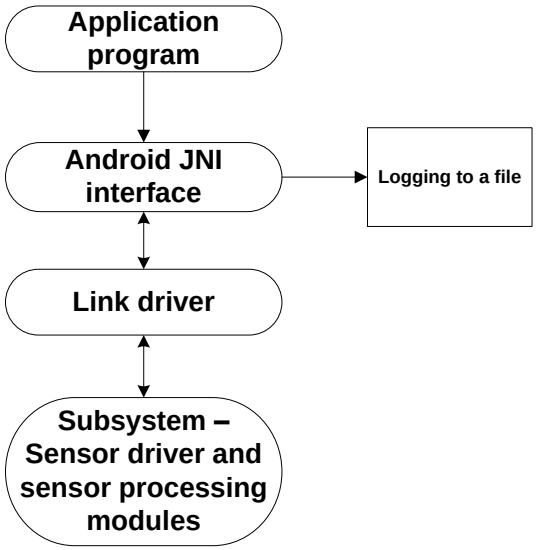


Figure 4.1: Test framework - High level design

The testing method is implemented in the application program along with the Bluetooth adaptor and host communication interface which runs on Android mobile

phone Samsung Galaxy S2 with Android OS version 2.3.4. The Android mobile device is connected to the Android SDK in laptop for installing/debugging application with the help of USB as shown in figure 4.2. The device chooser in Android SDK is used to select between the real mobile device and Android virtual device.

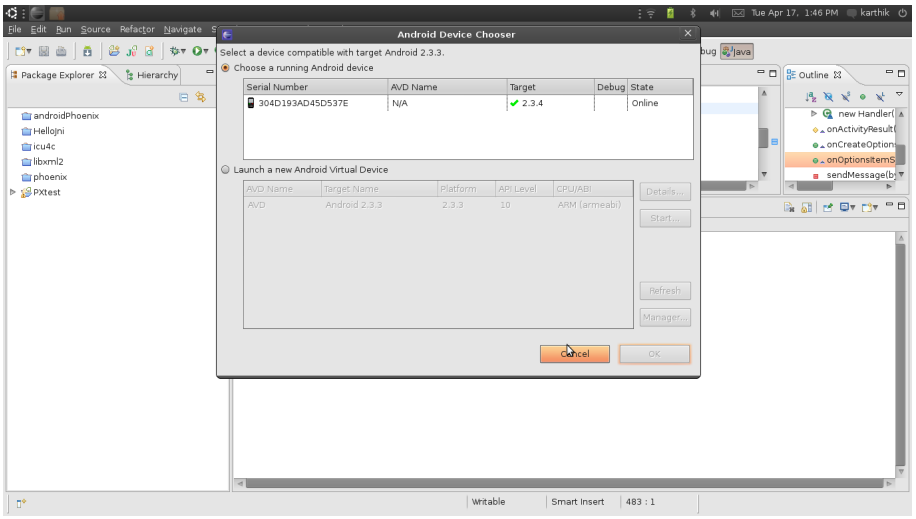


Figure 4.2: Android SDK - device manager

Test cases as shown in figure 4.4 are developed to test the individual functionality of the firmware and debug prints are added wherever necessary to find errors which can be seen in the debug console like shown in figure 4.3.

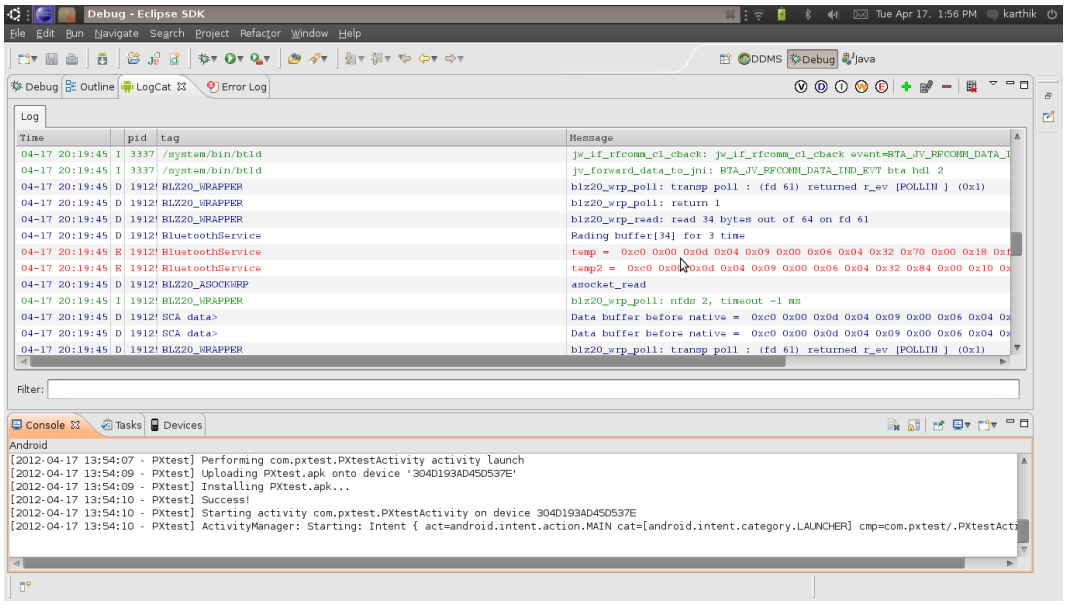


Figure 4.3: Android debug console - debug prints

Test cases are classified as sensor test cases and context algorithm test cases.

Sensor test cases: Polls sensor for a specified amount of time with specified sampling frequency and checks whether it returns any sensor data. Test case fails if

sensor does not return any value and also sensor output data rate not matched with the specified input sampling frequency. The test case result also displays number of sensor data produced and the specified sensor data rate are verified with the help of output time stamp information.

Accelerometer, Gyroscope and Magnetometer sensor data are tested in this test case. The difference between the successive time stamp information is proportional to the data rate which is used in run time environment to check the sensor data rate.

Thus this test case contains information such as name of the sensor, frequency and the required amount of time the sensor needs to be tested as shown in figure 4.4.

Context algorithm test case: This is an user based test case which means the user actions are continuously logged to a file for manual verification.

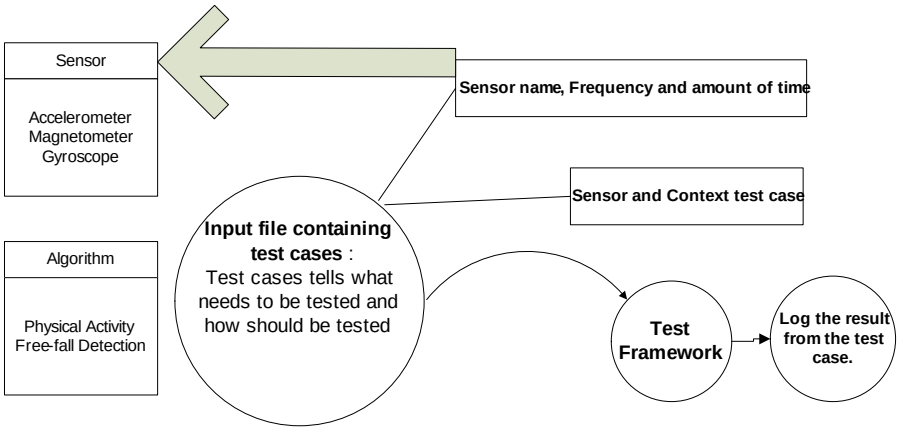


Figure 4.4: Test case - low level design

4.2 Testing the physical activity classification

The sensor subsystem is clocked at 30 MHz and thus requiring much less power to run the signal processing algorithm when compared to the main processor. The tri-axial accelerometer data coming from the sensor box are 16 bit signed integer having a dynamic range of $\pm 2G$ typical plots shown in figure 4.5. The accelerometer sensor data is collected for various human activities such as stand-still, walking, running, cycling, vehicle and sports using the application in the main processor. Data from 10 people are collected for training the activity context algorithm. This sensor data is used for training naive bayesian classifier algorithm in Matlab. Only useful part of the sensor data are used for training purpose and others like taking rest after running are not used for classification. Thus classifier is trained under a controlled lab environment.

Test samples containing data for all six activities are passed to the classification stage for detecting the activity. Figure 4.6 shows merged data of three axes of accelerometer using $\sqrt{x^2 + y^2 + z^2}$.

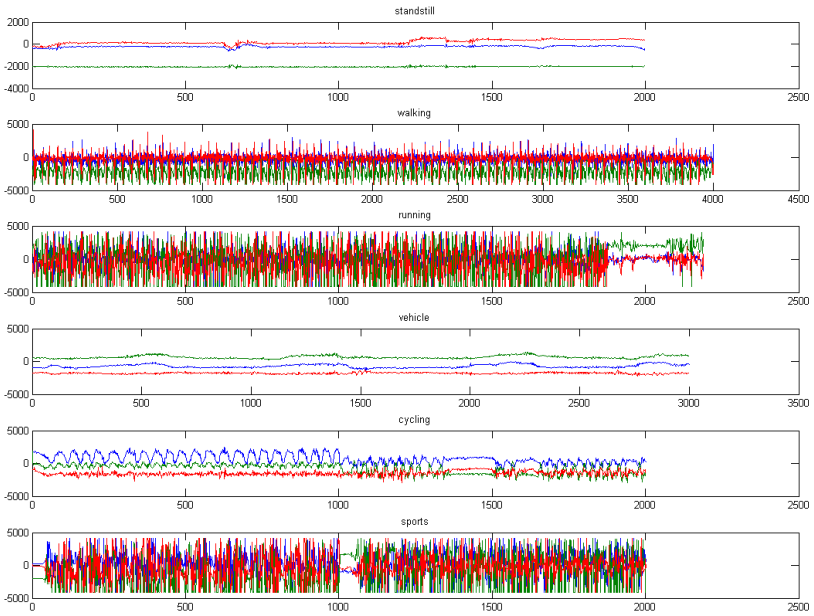


Figure 4.5: Raw sensor data

Standing and sitting involves no human movements so the frequency spectrum for these data is flat with no dominant frequency as shown in figure 4.7. Person sitting inside car or train also exhibit same behaviour as that of stand-still position figure 4.9. So the above two activity are difficult to classify as there is no user mobility which often leads to misclassification.

Running results in complex hip acceleration and most of the major frequency components lie between 0.5 Hz and 2Hz as seen in figure 4.8. Cycling involves an uniform circular movement of the legs, so a single dominant frequency component is seen around at 0.5 Hz due to vertical acceleration of the hip and a low magnitude for all other frequencies.

Ambulation and posture are similar across humans due to similar anatomy structure, but higher level activities such as playing floor-ball are more subject to personal behavioural patterns. Since sports involves sudden heavy running and stopping many high frequency component are present. Zero crossing rate is higher for higher level activities and lower for idle activities. Thus ZCR is a useful feature often used to classify human activities.

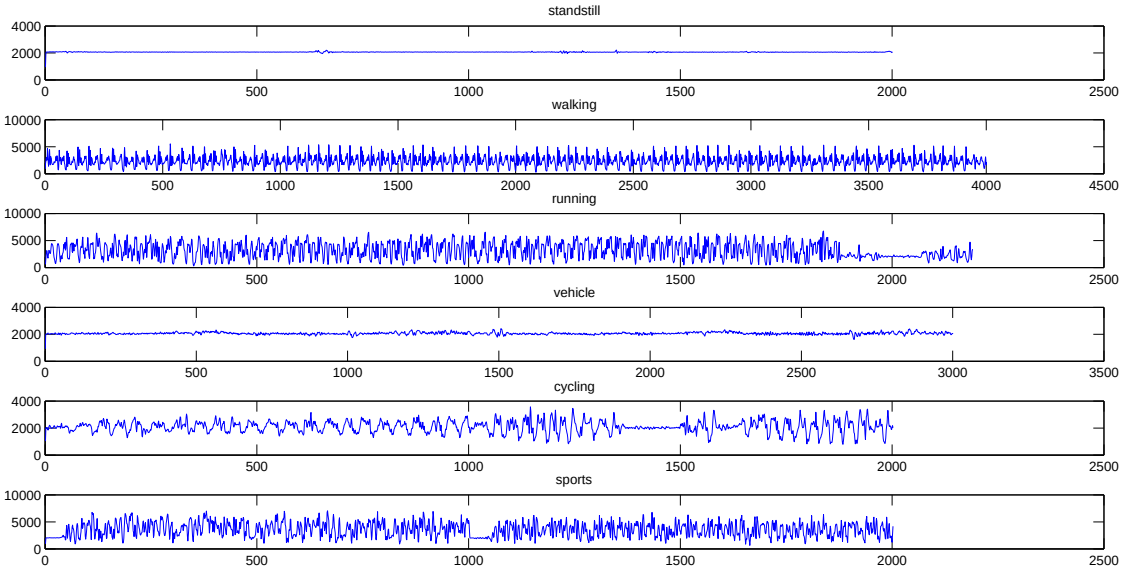
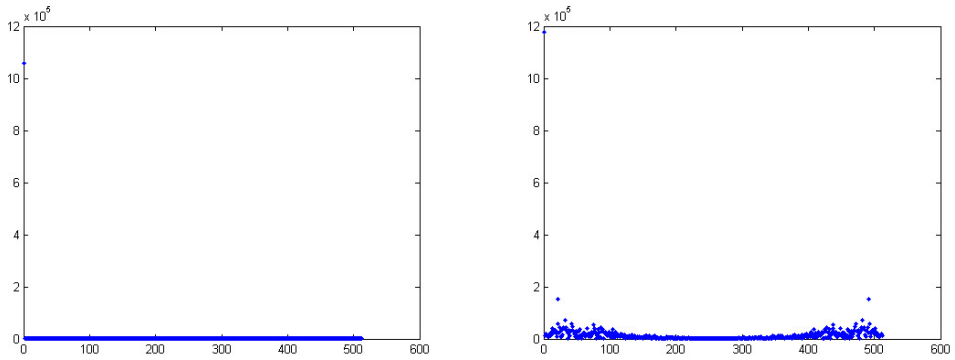


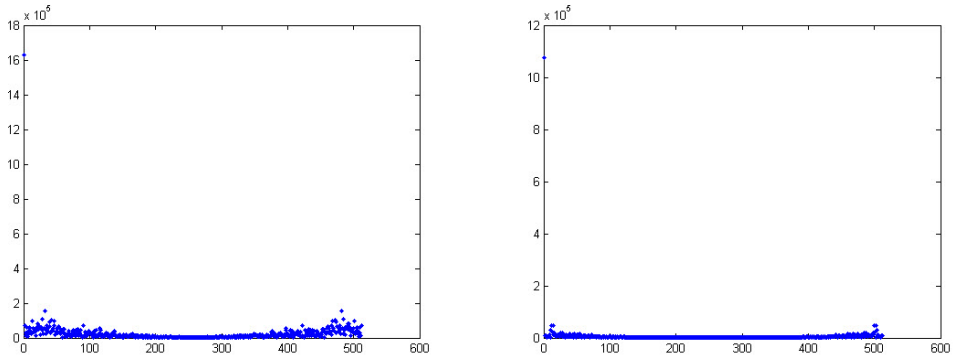
Figure 4.6: Merged sensor data



(a) Stand-still (b) Walking

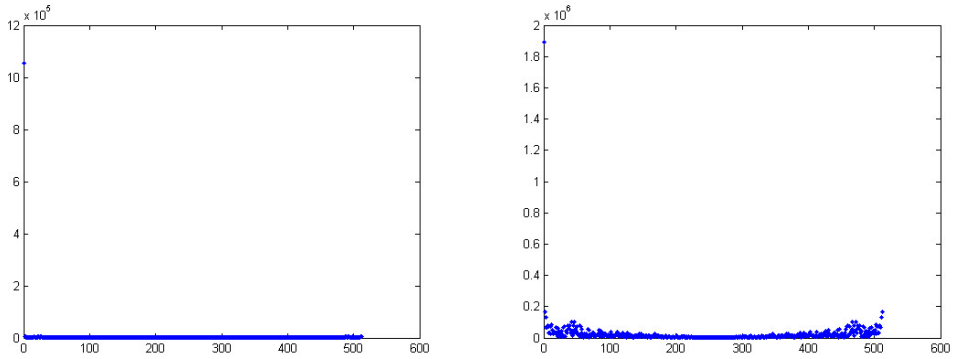
Figure 4.7: FFT plot for Stand-still and Walking activity

Confusion matrices resulting from physical activity testing is generated with the help of test sample which classifies the physical activity of the user is shown in table 4.1. Each activity is tested separately for 1000 iteration with random number used to generate the starting index and 512 samples are used in detection.



(a) Running (b) Bicycling

Figure 4.8: FFT plot for Running and Bicycling activity



(a) Vehicle (b) Sports

Figure 4.9: FFT plot for Vehicle and Sports activity

	Stand-still	**Walk**	**Run**	**cycle**	**Vehicle**	**Sports**	**Percentage**
Stand-still	819	0	0	0	181	0	81.9
Walk	0	931	0	69	0	0	93.1
Run	0	0	981	0	0	19	98.1
cycle	0	6	0	994	0	0	99.4
Vehicle	211	0	0	0	789	0	78.9
Sports	0	0	27	0	0	973	97.3

Table 4.1: Confusion matrix

Naive bayesian is a computationally efficient algorithm used for pattern classification purposes in a variety of applications. Due to the assumptions of conditional independence between features and normal distribution of feature values naive bayesian algorithm is unable to adequately model probability distributions. Further-

more, naive bayesian algorithm requires more data to accurately model probability distributions. Rule-based algorithm like decision trees captures connections in the feature vectors that lead to higher accuracy rate.

Although real-world activity recognition systems are relying on pre-trained classifiers on large data sets, individual training can lead to more accurate activity recognition. Pre-trained classifier offer greater convenience and easy to deploy in a short span of time. Nowadays on-line learning is getting popular where the user needs to tag the activity and train the classifier for sometime which can be used immediately after that. Thus particular user's behavioural movements are studied during the learning stage helping in accurately classifying the activities.

5. CONCLUSION

The low power continuous sensing architecture is implemented in the sensor hub whose architecture and implementation methodology are described in general term. It was shown that the dedicated sensor processor gives low latency feedback, the buffer capability and more processing of sensor data which reduces the interrupt rate to the application processor thereby reducing power consumption. The system framework can still be improved by dynamically loading the modules in the memory at run time rather than statically loading. With dynamic loading of the module, new module can be added to the application at run time instead of reinstalling the application again. System framework for mobile devices can also be improved by using publish/subscribe system service which is asynchronous, multicast and differs from traditional point to point model assuring reliable communication in dynamically changing environment.

Classification of physical activity of the user is implemented in Atmel ARM based microcontroller using fixed point C language by using the naive bayesian classifier algorithm. The classifier is trained using real time accelerometer data sample collected from the sensor box which is connected to the Android mobile via Bluetooth link. The performance of the classification system is verified in the simulation environment achieving a overall 92 percent accuracy.

Accuracy of the classifier can be improved by expanding the feature vector like calculating entropy, dot-product between axes to distinguish activity with similar energy level. Also collecting sensor data from multiple sensor box through Bluetooth placed around the body can detect the physical activity of the user more efficiently. Other machine learning algorithm can also be implemented in this platform in the future. Acceleration of the waist is the best for activity discrimination. This shows that an accelerometer attached to a mobile device, which is often placed at a fixed location such as on a waist or belt can recognize certain activities more efficiently.

Sensor hub also provides sensors drivers for gyroscope and magnetometer which can be used in the future for the development of navigation and orientation algorithms like Inertial Navigation System (INS), Inertial Motion Unit (IMU) and Magnetic, Angular Rate, and Gravity (MARG) sensor modules can be implemented and tested in this platform. Other algorithms like pedometer, screen orientation detection, etc can also be implemented in this platform.

REFERENCES

[1] Bodhi Priyantha, Dimitrios Lymberopoulos, Jie Liu "LittleRock: Enabling Energy-Efficient Continuous Sensing on Mobile Phones" *IEEE Pervasive Computing, vol.10, no.2, pp. 12 - 15*

[2] Macii, D., Boni, A., De Cecco, M., Petri, D. "multisensor data fusion" *Instrumentation Measurement Magazine, IEEE International Conference on June 2008, vol.11, issue.3, pp. 24-33*

[3] Luo, R.C., Kay, M.G. "A tutorial on multisensor integration and fusion" *Industrial Electronics Society, 1990. IECON '90., 16th Annual Conference of IEEE on 27-30 Nov 1990, vol.1, pp. 707-722*

[4] Dasarathy, B.V. "Sensor fusion potential exploitation-innovative architectures and illustrative applications" *Proceedings of the IEEE on Jan 1997, vol.85, issue.1, pp. 24-38*

[5] Grigera, J., Fortier, A., Rossi, G., Gordillo, S. "A Modular Architecture for Context Sensing" *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on 21-23 May 2007, vol.2, pp. 147-152*

[6] Ashbrook, D. "Context sensing with the Twiddler keyboard" *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on 1999, pp. 197-198*

[7] Ji-Sun Kim, Gracanin, D., Quek, F. "Sensor-fusion walking-in-place interaction technique using mobile devices" *Virtual Reality Workshops (VR), 2012 IEEE on 4-8 March 2012, pp. 39-42*

[8] A. Knoll, T. Christaller "Robotik" *Fischer Taschenbuch Verlag, Frankfurt, Germany, 2003*

[9] N. N. "Sensor" *http://de.wikipedia.org/wiki/Sensor (URL, Sept. 2005)*

[10] Sean C. Rhea "A Programmer's Tutorial on Event-Driven Programming, Asynchronous Input/Output, and the Bamboo DHT" *Proceedings of USENIX WORLDS 2005, December 2005*

[11] Mirco Musolesi, Mattia Piraccini, Kristof Fodor, Antonio Corradi, and Andrew T. Campbell "Supporting Energy-Efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones" *Pervasive Computing, 8th International Conference, Pervasive 2010, Helsinki, Finland, May 17-20, 2010*

[12] Minho Shin, Tsang, P., Kotz, D., Cornelius, C. "DEAMON: Energy-efficient Sensor Monitoring" *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on 22-26 June 2009, pp.1-9*

[13] Bodhi Priyantha, Dimitrios Lymberopoulos, and Jie Liu. "LittleRock: Enabing Energy Efficient Continuous Sensing on Moble Phones" *no. MSR-TR-2010-14, 18 February 2010*

[14] Henk Muller, Cliff Randell "An Event-Driven Sensor Architecture for Low Power Wearables" *ICSE 2000, Workshop on Software Engineering for Wearable and Pervasive Computing, pp. 39?41. June 2000*

[15] Huadong Wu "Sensor Data Fusion for Context-Aware Computing Using Dempster-Shafer Theory" *The Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213 December 2003*

[16] Hans-W. Gellersen, Albrecht Schmidt and Michael Beigl "There is more to Context than Location"*Proceedings of the International Workshop on Interactive Applications of Mobile Computing November 1998, Rostock, Germany*

[17] Hans-W. Gellersen, Albrecht Schmidt and Michael Beigl "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts" *Mobile Networks and Applications (MONET), Springer Netherlands, 2002, pp. 341-351*

[18] E.J. Selker and W. Burleson "Context Aware Design and Interaction in Computing Systems" *IBM Systems J., vol. 39, nos. 3-4, 2000, pp. 617-632*

[19] Peizhao Hu, Suan Khai Chong, Indulska, J., Krishnaswamy, S. "Context-aware and resource efficient sensing infrastructure for context-aware applications" *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on March 29 2010-April 2 2010, vol. 39, nos. 3-4, 2000, pp. 617-632*

[20] Nahla Ben Amor, Salem Benferhat, Zied Elouedi "Naive Bayes vs Decision Trees in Intrusion Detection Systems" *2004 ACM Symposium on Applied Computing*

[21] Youngjoo Suh , Youngjik Lee "Phoneme segmentation of continuous speech using multi-layer perceptron" *Electronics and Telecommunications Research Institute*

[22] Android Bluetooth class "http://developer.android.com/guide/topics/wireless/bluetooth.html"

[23] Android Bluetooth package
"http://developer.android.com/reference/android/bluetooth/package-summary.html"

[24] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Lee and Tae-Seoung Kim "A Triaxial Accelerometer-based Physical Activity Recognition via Augmented Signal Features and a Hierarchical Recognizer" *Information Technology in Biomedicine, IEEE Transactions on Sept. 2010, vol. 14, nos. 5 , pp. 1166-1172*

[25] Lafortune, M.A. Mehmood Khan, Young-Koo Lee, Sungyoung Lee and Tae-Seoung Kim "Three-dimensional acceleration of the tibia during walking and running" *Journal of Biomechanics, Vol. 24, No. 10, pp. 877?886*

[26] Yang, J.Y., Wang, J.S. and Chen, Y.P "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers" *Pattern Recognition Letters, Vol. 29, No. 16, pp. 2213-2220*

[27] Mathie, M.J., Coster, A.C.F., Lovell, N.H. Celler, B.G., "Detection of daily physical activities using a triaxial accelerometer" *Medical and Biological Engineering and Computing, Vol. 41, No. 3, pp. 296-301*

[28] Foerster, F. and Fahrenberg, J. "Motion pattern and posture: Correctly assessed by calibrated accelerometers" *Behavior Research Methods, Instruments, and Computers,Vol. 32, No. 3, pp. 450-456.*

[29] Htike, K.K., Khalifa, O.O., "Comparison of supervised and unsupervised learning classifiers for human posture recognition" *Computer and Communication Engineering (ICCCE), 2010 International Conference on 11-12 May 2010, pp. 1-6*

[30] Sun, M. Hill, J.O. "A method for measuring mechanical work and work efficiency during human activities" *Journal of Biomechanics, Vol. 26, No. 3, pp. 229-241*

[31] Bao, L. Intille, S.S. "Activity recognition from user-annotated acceleration data" *Proceedings of the 2nd International Conference on Pervasive Computing (Pervasive 2004). A. Ferscha M. Friedemann, (eds.). Springer, Berlin/Heidelberg,April 21-23, 2004. Pp. 1-17*

[32] JNI examples textif
http://android.wooyd.org/JNIExample/files/JNIExample.pdf

[33] Thread textif
http://developer.android.com/reference/java/lang/Thread.html

[34] SLIP textif
http://daxnetworks.com/Technology/TechDost/TD-061505.pdf

[35] Julian Stoev, KyuNam Cho, Jun-Seok Shim and Ho Seong Lee "Free Fall Detection Algotithms for Hard Disk Drives" *SICE-ICASE International Joint Conference 2006,18-21 Oct. 2006. pp.2760 - 2764*

[36] Fukaya, K. "Fall detection sensor for fall protection airbag" *SICE 2002. Proceedings of the 41st SICE Annual Conference,5-7 Aug. 2002. vol. 1, pp. 419-420*

[37] Oppenheim, A.V., Schafer, R.W. Buck, J.R. 1999. "Discrete-Time Signal Processing" *2nd edn, Prentice-Hall International*

[38] Cooley, J. Tukey, J.W. 1965 "An Algorithm for the Machine Computation of the Complex Fourier Series" *Mathematics of Computation, Vol. 19, pp. 297-301*

[39] Kevin P. Murphy "Naive Bayes classifiers" *October 24, 2006*

[40] Chen Jin "An improved ID3 decision tree algorithm" *Computer Science Education, 2009. ICCSE '09. 4th International Conference on, 25-28 July 2009. pp. 127 - 130*

[41] Jun-Hui Liu "Optimized ID3 algorithm based on attribute importance and convex function" *IT in Medicine and Education (ITME), 2011 International Symposium on, 9-11 Dec. 2011. vol. 2, pp. 136 - 139*

[42] Thakur, D. "Re optimization of ID3 and C4.5 decision tree" *Computer and Communication Technology (ICCCT), 2010 International Conference on, 17-19 Sept. 2010. pp. 448 - 450*

[43] Jia Wenguang "Improved C4.5 Decision Tree" *Internet Technology and Applications, 2010 International Conference on, 20-22 Aug. 2010. pp. 1 - 4*

[44] Lili Hao,Inst. of Math., Jilin Univ., Changchun "Temporal Data Driven Naive Bayesian Text Classifier" *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for, 18-21 Nov. 2008. pp. 699- 702*

[45] Alhammady, H. ,Etisalat Univ. Coll., Sharjah "Weighted Naive Bayesian Classifier" *Computer Systems and Applications, 2007. AICCSA '07. IEEE/ACS International Conference on, 13-16 May 2007. pp. 437-441*

[46] Ruck, D.W., Air Force Inst. of Technol., Wright-Patterson AFB, OH "Multisensor fusion classification with a multilayer perceptron" *Neural Networks, 1990., 1990 IJCNN International Joint Conference on, 17-21 Jun 1990. vol.2, pp. 863 - 868*

[47] Gas, B., Inst. of Intell. Syst. Robot., Pierre Marie Curie Univ., Paris, France "Self-Organizing MultiLayer Perceptron" *Neural Networks, IEEE Transactions on, Nov. 2010. Vol. 21, Issue. 11 , pp. 1766 - 1779*

[48] Hui Wang, Shanghai Jiao Tong Univ., Shanghai "Network Lifetime Optimization by Duality Approach for Multi-Source and Single-Sink Topology in Wireless Sensor Networks" *Communications, 2007. ICC '07. IEEE International Conference on 24-28 June 2007, pp. 3201 - 3206*

[49] Source, sink and filter concept
http://lua-users.org/wiki/FiltersSourcesAndSinks

[50] ATMEL at91sam7s256 datasheet
http://www.atmel.com/Images/6120s.pdf

A. APPENDIX

The drawbacks seen in naive-bayesian classifier which is discussed in the result section can be overcome with the help of rule based algorithm like decision trees. The classification accuracy can be improved by stripping the classification process with three states such as idle state, slow motion state and fast motion state with the help of custom based decision trees algorithm and later naive-bayesian classifier is used to classify physical activity from these states as shown in figure A.1. The advantages of naive-bayesian and decision trees are combined in this method while the disadvantages of both algorithms are avoided.

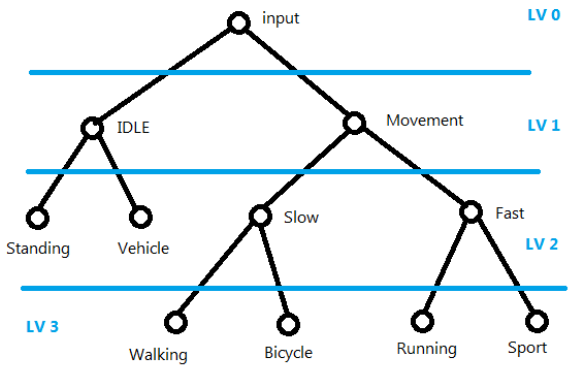


Figure A.1: Hybrid classifier using decision tree and naive-bayesian classifier algorithm

Feature vectors like zero crossing rate and peak frequency can be used to detect the correct state and then to classify the individual physical activity from these states, the different choice of feature vectors can be used. For example, dot-product can be used as a feature vector in idle state whereas frequency domain entropy can be used as a feature vector in slow movement state since they got similar energy level. Thus the complexity and accuracy of this algorithm needs to be studied in detail which can be taken as a future work.